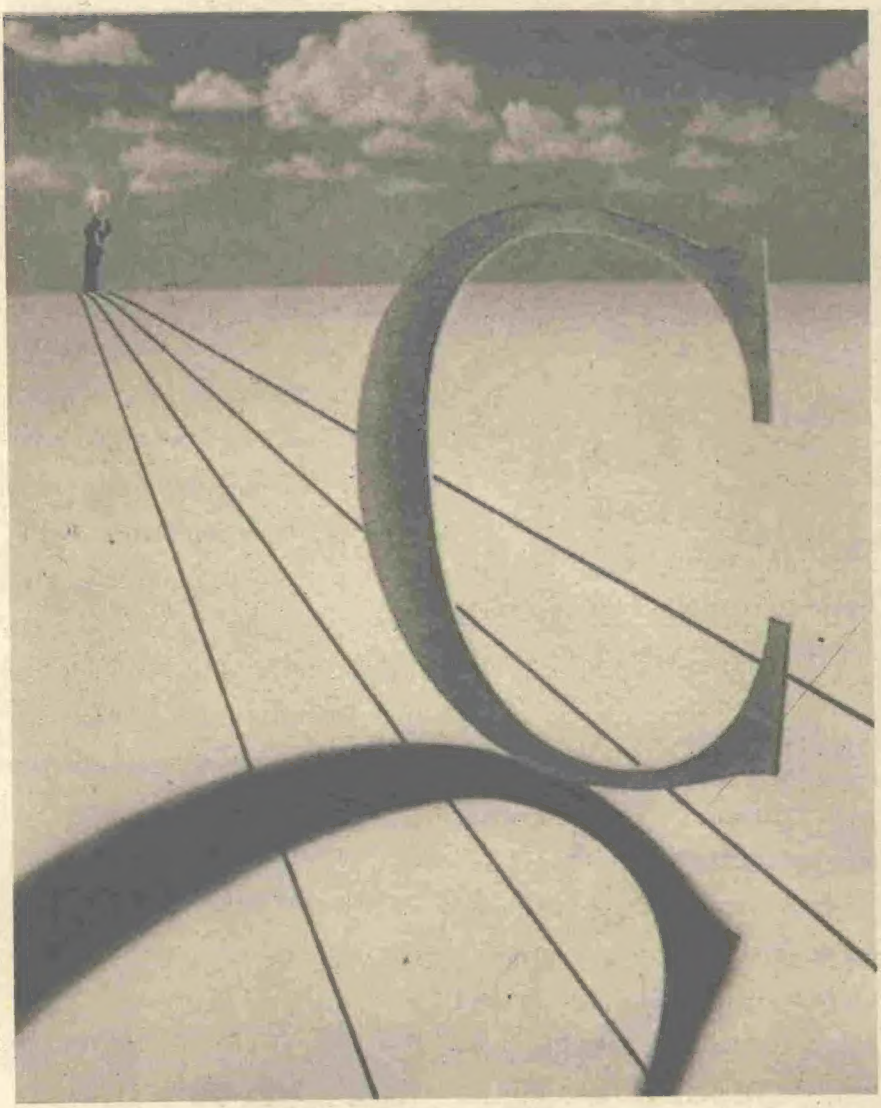
กิติพงษ์ ฤทธิบุตร

Integrated Debugger
นักสืบ Turbo C (1)

รายละเอียดหน้าที่ของคำสั่งต่าง ๆ ในการดีบักสำหรับผู้ใช้เทอร์โบซี 2.0



ถ้าคุณเป็นคนหนึ่งที่เคยเขียนโปรแกรมคอมพิวเตอร์คงจะรู้ดีว่า การเขียนโปรแกรมแต่ละโปรแกรมนั้นกว่าจะนำมาใช้งานได้จริงก็ต้องปลุกปล้ำกันนานพอควรทีเดียว น้อยคนนัก (คิดว่าไม่มีมากกว่า!) ที่สามารถเขียนโปรแกรมได้โดยไม่เกิดการผิดพลาดเลย ตั้งแต่เริ่มเขียนจนกระทั่งนำโปรแกรมมาใช้งานได้ เพราะโอกาสที่จะพบสิ่งขาดตกหกหล่นระหว่างขั้นตอนการเขียนโปรแกรมมีได้ตลอดเวลา แม้ว่าเราจะเริ่มจากขั้นตอนการคิดลำดับการทำงานของโปรแกรมก่อนที่จะลงมือเขียนโปรแกรมจริง ๆ เช่น อาจเขียนคำสั่งผิดพลาดทั้งในด้านโลจิกและซินแท็กซ์ของภาษาเอง ส่งผลไปถึงขั้นตอนการแปล (compile) ที่จะต้องแก้ไขกันอีกจนในที่สุดเมื่อสามารถผ่านขบวนการคอมพายล์ไปได้ -และสามารถเข้าสู่ขั้นตอนการรัน ซึ่งในขั้นตอนนี้อาจจะมีข้อผิดพลาดเกิดขึ้นอย่างที่เรียกว่า runtime error

เมื่อตรวจสอบแก้ไขจนผ่านขั้นนี้ไปได้ (โปรแกรมสามารถรันได้โดยไม่มีความผิดพลาดเกิดขึ้นระหว่างการประมวลผล) ก็มิใช่ว่าจะประสบความสำเร็จ เพราะการทำงานของโปรแกรมอาจไม่ให้ผลลัพธ์ (output) ตามที่ต้องการ เราเรียกว่าโปรแกรมมี "บัก" ถ้าเกิดเหตุการณ์อย่างนี้ขึ้นจริงก็คงจะต้องหาจุดที่ผิดพลาดและแก้ไขกันต่อไป นั่นคือการดีบัก ซึ่งเป็นสิ่งที่สร้างความหนักใจให้กับโปรแกรมเมอร์ทั้งหลาย เพราะเขาต้องเสียเวลาไล่เรียงการทำงานของแต่ละคำสั่งในโปรแกรมตามลำดับ จึงไม่ใช่งานเล็กน้อยแน่นอน โดยเฉพาะอย่างยิ่งพวกโปรแกรมขนาดใหญ่ที่มีการทำงานซับซ้อน การตรวจสอบหา

จุดผิดพลาดและแก้ไขให้ถูกต้องนั้นเป็นงานที่ชวนให้ "ปวดหมอง" มากที่สุด เครื่องมือที่ช่วยแก้ไขได้ทางหนึ่งคือ "ดีบักเกอร์" โปรแกรมชุดหนึ่งที่มีขีดความสามารถในการกำหนดขั้นตอนการประมวลผลโปรแกรม หรือทดสอบการทำงานของชุดคำสั่งต่าง ๆ เพื่อช่วยตรวจสอบหาจุดผิดพลาดในโปรแกรมคอมพิวเตอร์ ผู้เขียนจะขอกล่าวถึง Integrated Debugger ที่มีให้ในเทอร์โบ-ซีเวอร์ชัน Integrated Environment

รู้จักกับ
Integrated Debugger

สำหรับดีบักเกอร์ของเทอร์โบซีตัวนี้เป็นดีบักเกอร์ในระดับภาษาต้นฉบับ (source-level) หมายความว่าในการใช้งานผู้ใช้สามารถควบคุมได้ด้วยการใช้ภาษาเดียวกันกับที่ใช้เขียนโปรแกรมหรือภาษาซีที่ใช้กับเทอร์โบซีนั่นเอง เช่น ผู้ใช้อาจสั่งให้ระบบแสดงค่าของตัวแปรตัวใดตัวหนึ่งในอะเรย์ได้ด้วยการใช้คำสั่ง rptr->image [nptr + 0×80]

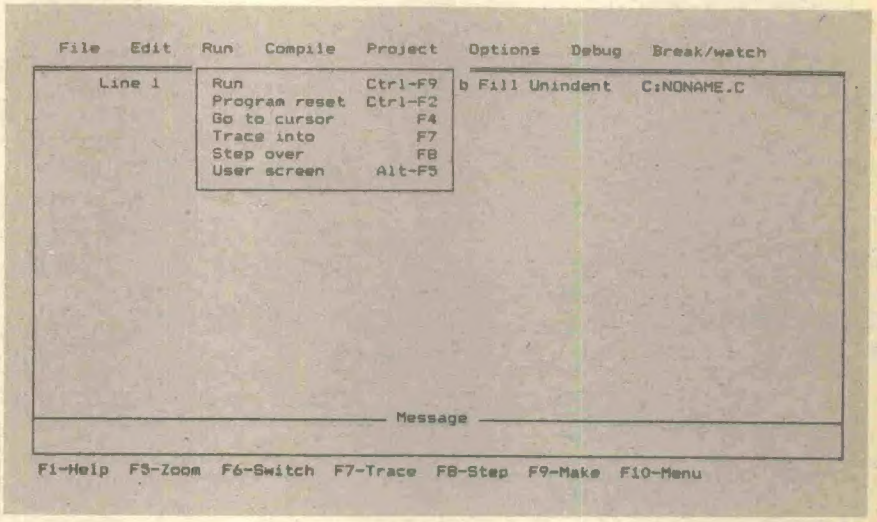
วิธีการที่จะดีบักโปรแกรมด้วยดีบักเกอร์เริ่มจากการกำหนดค่าให้ source debugging มีค่าเป็น On โดยเลือกเมนู Debug/Source debugging แล้วตรวจสอบการทำงานของโปรแกรมที่ต้องการผ่านเมนู Run/Run หรือใช้แป้น Ctrl-F9 ฮอตคีย์สำหรับคำสั่งนี้ แต่ก่อนที่จะทำการรันโปรแกรม ผู้ใช้สามารถกำหนดตำแหน่งที่ต้องการให้ระบบหยุดการรันโปรแกรม (breakpoint) ที่จุดไหนเป็นจำนวนกี่แห่งไว้ในซอร์สไฟล์ได้ตามที่ต้องการ เมื่อสั่งให้รันโปรแกรมแล้ว ระบบจะหยุดเมื่อถึงตำแหน่งที่ระบุและคืนการควบคุมให้กับผู้ใช้เพื่อทำการดีบักโปรแกรมต่อไป ผู้ใช้สามารถพิจารณาและจัดการกับโปรแกรมที่กำลังตรวจสอบอยู่นั้นได้หลายอย่างด้วยกัน คือ

- ให้แสดงค่าของตัวแปรหรือ expression ที่ต้องการ
- กำหนดรูปแบบของ expression หลาย ๆ ชุดไว้ในกรอบ (special window) ที่ระบบจัดเตรียมไว้ให้ เพื่อตรวจสอบค่าที่ได้จากการประมวลผล expression เหล่านั้น
- เปลี่ยนแปลงค่าของตัวแปรในโปรแกรม
- ยกเลิกตำแหน่งที่กำหนดไว้เป็น breakpoint หรือกำหนดตำแหน่งที่ต้องการให้หยุดในจุดอื่นต่อไป
- สั่งให้ระบบทำการรันโปรแกรมครั้งละหนึ่งบรรทัด
- แก้ไขซอร์สไฟล์และทำการแปลใหม่ (recompile) รวมถึงการลิงก์โปรแกรมเพื่อทำการรันใหม่ อาจใช้คำสั่งอะไรก็ได้ที่มีอยู่ในเมนูของเทอร์โบซี
- สั่งให้รันโปรแกรมต่อไปจนกว่าจะพบ breakpoint แห่งใหม่

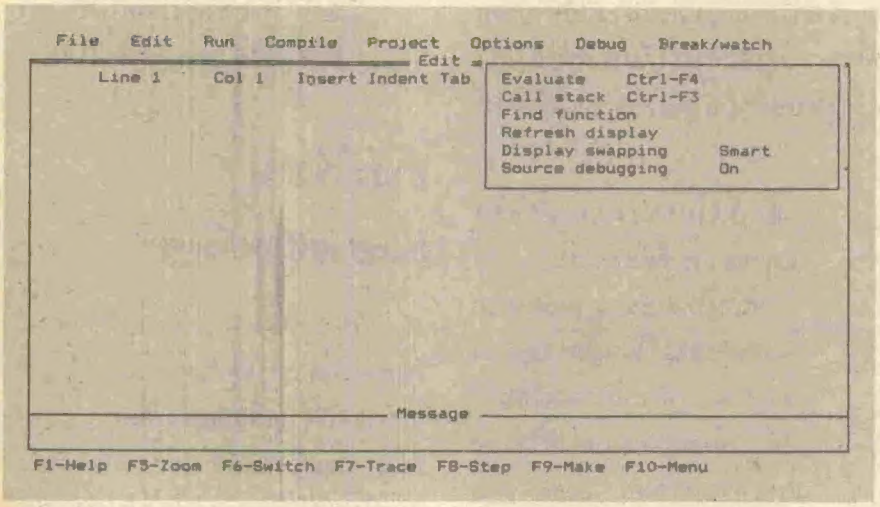
คำสั่งใช้งาน
Integrated Debugger

ในการใช้งานโปรแกรมดีบักเกอร์เราควรทำความรู้จักกับชุดคำสั่งต่าง ๆ ที่จะนำมาใช้งานเสียก่อน เพื่อสร้างความคุ้นเคยและเรียนรู้วิธีการสั่งงาน คำสั่งส่วนใหญ่ที่ใช้ในการดีบักโปรแกรมนั้นแทบจะครอบคลุมชุดคำสั่งของเทอร์โบซีทั้งหมด เพราะผู้ใช้สามารถกระทำสิ่งใดในขณะดีบักได้ แต่คำสั่งที่ใช้งานกับดีบักเกอร์จริง ๆ หรือจำเป็นต้องใช้บ่อยครั้ง ได้แก่ คำสั่งต่าง ๆ ในเมนู Run, Debug, Options และ Break/Watch ดังรูปที่ 1, 2, 3 และ 4 แต่ละคำสั่งจะทำหน้าที่แตกต่างกันดังนี้

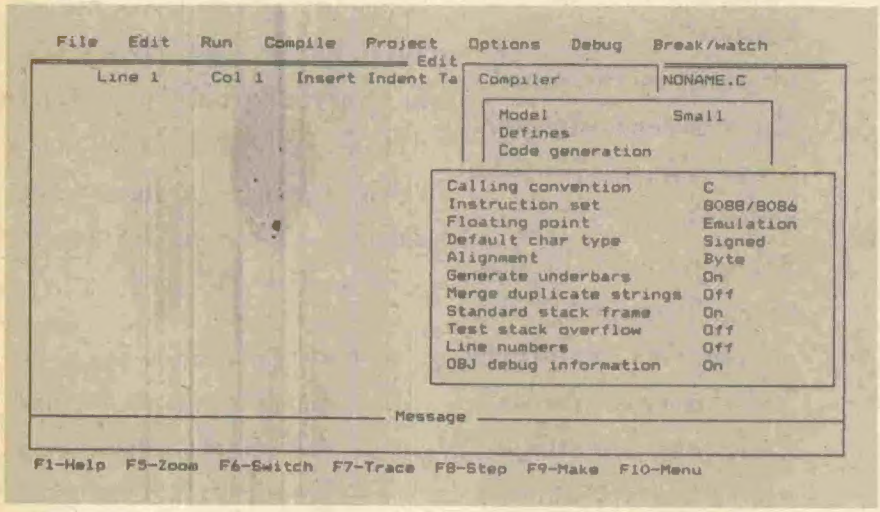
- Run/Goto cursor มีแป้น F4 เป็นฮอตคีย์ของคำสั่ง คำสั่งนี้จะสั่งให้ระบบทำการรันโปรแกรมไปจนถึงบรรทัดที่เป็นตำแหน่งของเคอร์เซอร์ ช่วยให้ผู้ใช้



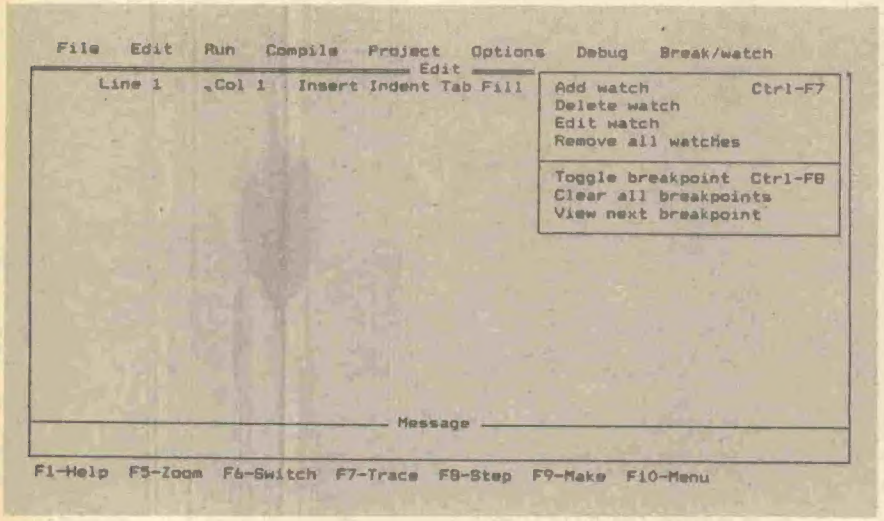
รูปที่ 1 กลุ่มคำสั่งในเมนู Run



รูปที่ 2 กลุ่มคำสั่งในเมนู Debug



รูปที่ 3 กลุ่มคำสั่งในเมนู Option



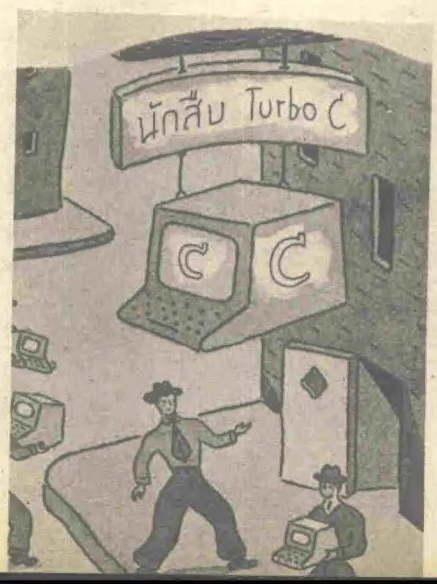
รูปที่ 4 กลุ่มคำสั่งในเมนู Break/watch

สามารถกำหนดส่วนของโปรแกรมที่ต้องการดีบักได้ทางหนึ่ง

- Run/Program reset เป็นคำสั่งยกเลิกการใช้พื้นที่หน่วยความจำที่ได้จองไว้ และทำการปิดไฟล์ต่าง ๆ ที่ใช้งาน การใช้คำสั่งนี้อาจสั่งด้วย Ctrl-F2 ซึ่งเป็นฮอตคีย์ได้เช่นกัน
- Run/Trace into เป็นคำสั่งที่ใช้ฟังก์ชันคีย์หมายเลข 7 (F7) เป็นฮอตคีย์ โดยจะทำหน้าที่รันคำสั่งชุดต่อไปในฟังก์ชันที่ตรวจสอบในขณะนั้น เช่น ถ้ามีการเรียกใช้ฟังก์ชันประเภท low-level function ที่ถูกคอมพายล์ด้วยการกำหนดค่า ให้ใช้คำสั่ง Debug/Source debugging และ O/C/C/Obj debug information เป็น On แล้วติดตามเข้าไปดูวิธีการทำงานของฟังก์ชันนั้นด้วย
- Run/Step over ทำงานเช่นเดียวกับ Run/Trace into แต่จะไม่เข้าไปทดสอบการทำงานภายในของฟังก์ชันที่ถูกเรียกใช้ต่อจากฟังก์ชันที่กำลังตรวจสอบ ส่วนฮอตคีย์ของคำสั่งนี้คือ F8
- O/C/C/Standard stack frame เป็นคำสั่งเพื่อให้การเรียกใช้สแต็กของระบบเป็นไปอย่างถูกต้อง โดยที่คำสั่ง Options/Compiler/Code generation/Standard stack frame จะต้องถูกกำหนดค่าให้เป็น On เสมอ.
- O/C/C/Obj debug information เป็นคำสั่งสำคัญอีกคำสั่งหนึ่งสำหรับเรียกให้ระบบเก็บข้อมูลที่จำเป็นต่อการดีบักเอาไว้ด้วย

โดยจะต้องกำหนดค่าให้คำสั่งนี้เป็น On เสมอถ้าต้องการให้ซอร์สไฟล์ที่ใช้คอมพายล์สามารถนำมาดีบักได้

- Debug/Evaluate เป็นคำสั่งที่ใช้เพื่อให้ระบบทำการประมวลผล expression ที่ต้องการ เป็นคำสั่งที่ผู้ใช้สามารถเปลี่ยนแปลงแก้ไขค่าของตัวแปรต่าง ๆ ในโปรแกรมได้ ส่วนวิธีการใช้งานคำสั่งนอกจากจะใช้ระบบเมนูแล้วยังสามารถใช้แป้นฮอตคีย์ Ctrl-F4
- Debug/Find function ทำหน้าที่สั่งให้ระบบแสดงโครงสร้างการทำงานของฟังก์ชันที่ต้องการ ซึ่งระบบจะแสดงให้เห็นใน Edit window
- Debug/Call stack สามารถสั่งงานด้วยการกดแป้น Ctrl-F3 (ฮอตคีย์) เพื่อแสดงการใช้งานสแต็ก โดยให้ระบบแสดงชุดคำสั่งของฟังก์ชันที่กำลังตรวจสอบขณะนั้นได้ด้วยการเลือกชื่อฟังก์ชันจากสแต็ก
- Debug/Source debugging คำสั่งนี้จะต้องถูกกำหนดค่าให้เป็น On เสมอถ้าต้องการที่จะดีบักโปรแกรม และสามารถเลือกได้ทั้งแบบ integrated และ standalone หากกำหนดให้เป็นแบบ standalone โปรแกรมที่ได้จะสามารถใช้กับ standalone debugger เพียงอย่างเดียว ถึงแม้จะยังคงนำมารันในเทอร์โบซีได้ตามปกติ เมื่อคำสั่งนี้ถูกกำหนดค่าให้เป็น None จะทำให้ไฟล์ที่มี extension เป็น .EXE ที่ได้ไม่มีข้อมูลสำหรับการดีบัก (debugging information) อยู่ด้วย ซึ่งส่งผลให้ผู้ใช้ไม่สามารถดีบักโปรแกรมนั้นได้ไม่ว่าจะใช้ดีบักเกอร์ตัวไหน
- Break/Watch/Add watch มี Ctrl-F7 เป็นฮอตคีย์ ทำหน้าที่เพิ่ม Watch expression
- Break/Watch/Delete watch เป็นคำสั่งที่ทำงานตรงกันข้ามกับ Add watch คือลบ Watch expression
- Break/Watch/Edit watch จะช่วยให้ผู้ใช้สามารถแก้ไข Watch expression ต่าง ๆ ได้
- Break/Watch/Remove all watches ทำงานเหมือนกับ Delete watch คือการลบ Watch expression แต่จะต่างกันตรงที่เป็นการลบทุก expression ออกทั้งหมด
- Break/Watch/Toggle breakpoint ใช้ในการกำหนดหรือยกเลิกตำแหน่งของ breakpoint ที่อยู่ในตำแหน่งของเคอร์เซอร์ขณะนั้น ซึ่งผู้ใช้สามารถกดแป้น Ctrl-F8 หรือฮอตคีย์ของคำสั่งนี้ในการเรียกใช้งานได้



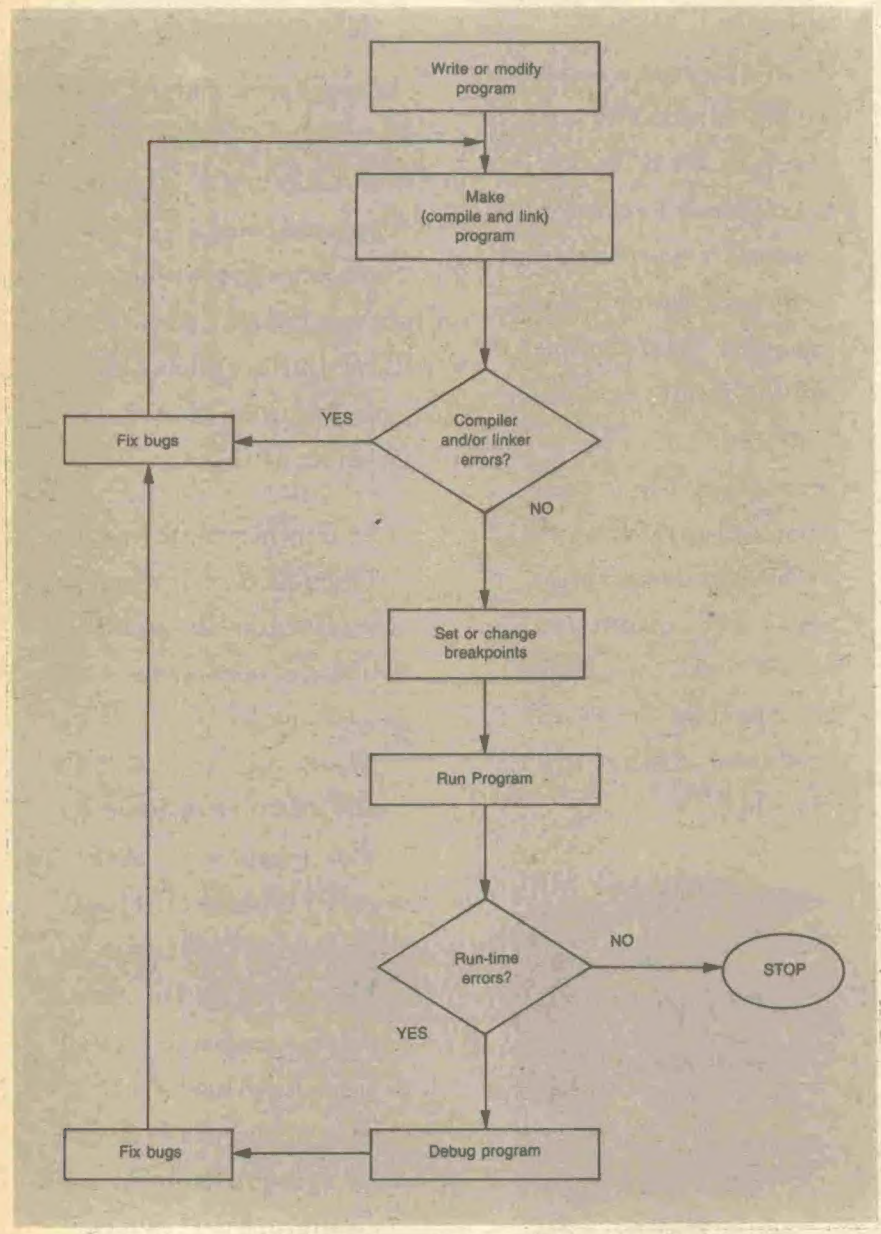
- Break/Watch/Clear breakpoint เป็นคำสั่งยกเลิกตำแหน่งของ breakpoint ที่อยู่ในตำแหน่งของเคอร์เซอร์ขณะนั้นด้วยปุ่มฮอตคีย์คือ Ctrl-F8
- Break/Watch/Clear breakpoint ยกเลิกตำแหน่ง breakpoint ทั้งหมดที่อยู่ในโปรแกรม
- Break/Watch/Next breakpoint เป็นคำสั่งเพื่อให้ระบบแสดงตำแหน่งของ breakpoint อันดับต่อไป

นอกเหนือจากคำสั่งต่าง ๆ ที่ใช้กับดีบักเกอร์โดยตรงแล้ว ยังมีคำสั่งอื่นที่ใช้กันบ่อยครั้งในการดีบักโปรแกรม (เมื่อใช้ Integrated Debugger) ส่วนใหญ่เป็นคำสั่งเกี่ยวกับการใช้จอภาพเพื่อแสดงผลในหลาย ๆ ลักษณะ ผู้ใช้สามารถนำไปใช้ได้ในทุกโอกาสขณะที่ใช้งานเทอร์โบซี ไม่ว่าจะอยู่ในขั้นตอนของการดีบักโปรแกรมหรือไม่

- F5 เป็นคำสั่งฮอตคีย์ของระบบมีหน้าที่ดึง (zoom) หรือขยายขนาดจอภาพของหน้าต่างที่กำลังทำงานจนเต็มจอ (full screen mode) หรือทำให้กลับสู่ลักษณะเดิม (unzoom) คือแสดงการแบ่งจอภาพออกเป็นส่วน ๆ (split-screen mode)
- Alt-F5 เป็นฮอตคีย์ที่ใช้ในการสลับจอภาพจากเทอร์โบซีสกรีนให้เป็นยูสเซอร์สกรีน และจะกลับคืน integrated environment (เทอร์โบซีสกรีน) ได้ด้วยการกดแป้นพิมพ์อะไรก็ได้
- F6 คำสั่งแบบฮอตคีย์ที่ทำงานเกี่ยวกับการบังคับการแสดงผลบนจอภาพ โดยจะทำหน้าที่โอน

ย้ายการควบคุมจอภาพ (active window) ไปมาระหว่าง Edit window และ Watch หรือ Message window

- Alt-F6 เป็นฮอตคีย์ที่ทำหน้าที่สองอย่างคือ ถ้าใช้คำสั่งนี้ในขณะที่ทำงานอยู่ใน Edit window จะเป็นการสั่งให้ระบบเรียกใช้ไฟล์ที่ถูกดึง (load) เข้ามาให้เอดิเตอร์เป็นอันดับสุดท้าย แต่ถ้าเรียกใช้คำสั่งนี้ขณะที่ทำงานในจอภาพส่วนล่าง (Watch/Message window) เป็นการสั่งให้ระบบสลับการทำงานระหว่าง Watch window และ Message window
- Run/Run เป็นคำสั่งในรันเมนูทำหน้าที่รันโปรแกรม โดยอาจจะใช้คำสั่งนี้เพื่อรันโปรแกรมตามปกติหรือร่วมกับดีบักเกอร์ก็ได้ ซึ่งก่อนที่จะใช้คำสั่งนี้ได้ก็จะต้องผ่านขั้นตอนการคอมพายล์และการลิงก์มาก่อนจนได้ไฟล์ที่มี extension เป็น .EXE แล้ว และถ้าโปรแกรมที่ได้ถูกคอมพายล์โดยการกำหนดค่าของ Debug/Source debugging และ O/C/C/Obj debug information เป็น On แล้วทำให้เราสามารถรันโปรแกรมเพื่อทำการดีบักได้ตามต้องการ
- Project/Remove messages เป็นอีกคำสั่งหนึ่งที่ใช้ในการลบข้อความต่าง ๆ ภายใน Message window



รูปที่ 5 แผนภาพแสดงขั้นตอนการดีบักโปรแกรม

ขั้นตอนการดีบักโปรแกรม

ก่อนที่เราจะใช้เทอร์โบซีดีบักเกอร์ (Integrated Debugger) เราลองมาดูขั้นตอนการดีบักโปรแกรมอย่างคร่าว ๆ ก่อนว่า จะมีรูปแบบหรือวิธีการทำงานอย่างไร แล้วเราค่อยศึกษาตัวอย่างการดีบักโปรแกรมด้วย Integrated Debugger กัน คงจะต้องเริ่มตั้งแต่การเขียนโปรแกรมเลย เสร็จแล้วก็ทำการคอมพายล์และลิงก์โปรแกรมนั้น คอยดูผลจากการคอมพายล์ว่ามี error ไหม ถ้ามีก็ทำการแก้ไขก่อนกลับไปทำการคอมพายล์ใหม่ ทำขั้นตอนนี้ซ้ำอีกจนไม่มี error เกิดขึ้น จะได้ไฟล์ที่มี extension เป็น .EXE ซึ่งสามารถนำไปรันได้ จากนั้นค่อยทำการกำหนดตำแหน่งที่จะให้ระบบหยุดทำการรันโปรแกรมในจุดที่ต้องการ เพื่อดูแลการทำงานของโปรแกรมเป็นส่วน ๆ รันโปรแกรมต่อไปเพื่อดูว่ามี run time error เกิดขึ้นหรือไม่ ถ้ามีต้องแก้ไขโปรแกรม แล้วทำการคอมพายล์ใหม่อีกครั้ง จนถึง

ขั้นตอนตรวจสอบว่าไม่มีการเกิด run-time error ก็เป็นอันว่าจบสิ้นขั้นตอนการดีบักโปรแกรม ขั้นตอนการดีบักโปรแกรมที่เรากล่าวถึงนี้ได้แสดงไว้ในรูปที่ 5

สำหรับการใช้งาน Integrated Debugger เราจะใช้โปรแกรมตัวอย่างที่บริษัทบอร์แลนด์อิงก์ให้มาพร้อมกับแพคเกจของเทอร์โบซี เวอร์ชัน 2.0 เพราะฉะนั้นเราจะข้ามขั้นตอนการเขียนโปรแกรมไปสู่ขั้นตอนการดีบักเลย ซึ่งตัวอย่างที่นำมาใช้ในการดีบักโปรแกรมนี้เริ่มจากการหาบักโดยใช้วิธีการง่าย ๆ ไปจนถึงการดีบักที่ต้องใช้วิธีการซับซ้อนยิ่งขึ้นเรื่อย ๆ

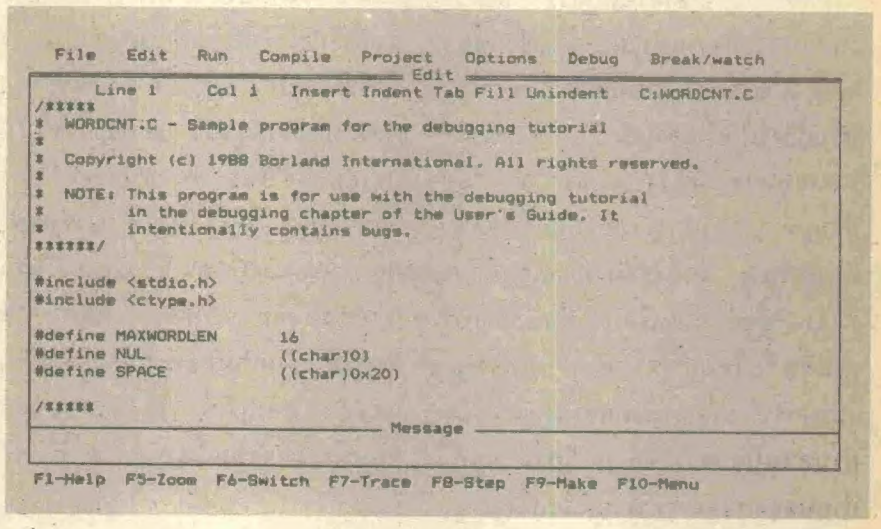
ลองดีบักโปรแกรมกัน

โปรแกรมที่เราจะนำมาเป็นตัวอย่างในการใช้ Integrated Debugger ค้นหาบักจากโปรแกรมก็คือ WORDCNT.C ซึ่งเป็นโปรแกรมที่จะทำหน้าที่แสดงส่วนประกอบของเทกซ์ไฟล์ และนับความยาวของคำ แล้วแสดงรายงานจำนวนคำที่มีความยาว 1 ตัวอักษร หรือคำที่มีความยาว 2 ตัวอักษรไปจนถึงคำที่มีความยาวมากที่สุด ทางบริษัทบอร์แลนด์ได้สร้างบักไว้หลายแห่ง เพื่อใช้เป็นตัวอย่างในการหาบักโปรแกรม โดยได้แสดงซอร์สโค้ดไว้ให้ด้วย เผื่อว่าถ้าเราได้โปรแกรมนี้มาจากแพคเกจเทอร์โบซีที่ได้มีการเปลี่ยนแปลงแก้ไขไฟล์นี้แล้ว จะได้นำมาตรวจสอบและแก้ไขให้เหมือนต้นฉบับ เมื่อนำมาทดลองจะได้ไม่เกิดความสับสน สำหรับโปรแกรมที่เราจะใช้งานมีชุดโปรแกรมของเทอร์โบซีในส่วนที่เป็น Integrated Environment, WORDCNT.C, WORDCNT.PRJ และ WORDCNT.DAT ซึ่งจะใช้เป็นไฟล์ข้อมูลของโปรแกรม WORDCNT.EXE โดยที่ไฟล์ทั้งหมดควรจะอยู่ในไดเรคทอรีเดียวกันเพื่อความสะดวกในการใช้งาน เมื่อทุกอย่างพร้อมก็เริ่มงานกันได้เลยด้วยการพิมพ์คำสั่งที่เครื่องหมายเตรียมพร้อมของดอส

```
TC WORDCNT (Enter)
```

หรือเรียกใช้เทอร์โบซีก่อน แล้วค่อยใช้คำสั่ง File จากเมนูหลักเพื่อสั่งให้เทอร์โบซีดึงไฟล์ WORDCNT.C เข้ามาในระบบด้วยคำสั่ง Load หรือกดแป้น F3 ระบบจะแสดงกรอบให้ใส่ชื่อไฟล์ที่ต้องการ ตอนนี้หน้าจอของเราจะเป็น TC screen พร้อมกับซอร์สโค้ดของโปรแกรม WORDCNT.C ใน Edit window ดังรูปที่ 6 เลือกใช้คำสั่ง Compile/Build all (เมนูบาร์หรือกดแป้นตัวอักษรใหญ่ของชื่อคำสั่งที่ต้องการคือ C และ B) สร้างโปรแกรมเทอร์โบซีจะทำการคอมพายล์และลิงก์โปรแกรม WORDCNT ให้เป็นไฟล์ .EXE ที่สามารถนำไปประมวลผลต่อไป สำหรับการสร้างโปรแกรม WORDCNT.EXE นี้จะต้องแน่ใจว่าคำสั่ง Debug/Source debugging และ O/C/C/Obj debug information ถูกกำหนดค่าไว้เป็น On (ซึ่งปกติทั้งสองคำสั่งนี้มีค่าดีฟอลต์เป็น On อยู่แล้ว) เมื่อระบบสร้างโปรแกรม WORDCNT.EXE เรียบร้อยและหยุดรอคำสั่ง เราจะเลือกใช้คำสั่ง Run/Trace into หรือใช้แป้น F7 ซึ่งเป็นฮอตคีย์ก็ได้ เพื่อสั่งรันโปรแกรม WORDCNT ด้วยดีบักเกอร์

เมื่อเราเรียกดีบักเกอร์ออกมาทำงาน เราต้องเลื่อนจอภาพของ Edit window ไปจนถึงตำแหน่งเริ่มต้นของฟังก์ชัน main () โดยจะหยุดรอที่บรรทัดแรกของฟังก์ชัน และแสดงแถบไฮไลต์ที่บรรทัดนี้ด้วย ในโปรแกรมตัวอย่างดีบักเกอร์จะไปหยุดอยู่ที่บรรทัด void main (void) บรรทัดนี้เป็นบรรทัดแรกที่ถูกประมวลผลเมื่อทำการรันโปรแกรม WORDCNT สำหรับบรรทัดที่มีไฮไลต์จะบอกให้รู้ว่าคำสั่งชุดต่อไปที่ได้รับการประมวลผลคือคำสั่งชุดไหน (execution position) เพื่อให้โปรแกรม WORDCNT ทำงานต่อไป เราต้องเลือกคำสั่ง Run/Run โปรแกรมจะเริ่มทำงานและหยุดรอให้ใส่ชื่ออินพุตไฟล์ เราก็พิมพ์ชื่อ



รูปที่ 6 โปรแกรมตัวอย่าง WORDCNT.C ที่ใช้ทดสอบการดีบัก

WORDCNT.DAT ที่เป็นไฟล์ข้อมูลการใช้งานในครั้งนี้ และกดแป้น Enter ทำให้ WORDCNT ทำงานต่อโดยอ่านและแสดงข้อมูลบรรทัดแรกของดาต้าไฟล์ ระบบจะหยุดทำงาน (lock up) ที่จุดนี้เพราะพบบักตัวโตที่บอร์แลนด์ใส่ไว้ในโปรแกรม ถึงตรงนี้เราจำเป็นต้องกดแป้น Ctrl-Break เป็นการสั่งให้ระบบปลดล็อค แล้วกดแป้น Esc เพื่อทดสอบว่าสามารถทำงานต่อได้ จากนั้นก็เลือกคำสั่ง Run/Program reset หรือกดแป้น Ctrl-F2 ฮอตคีย์ที่สั่งให้หยุดการทำงานของดีบักเกอร์ ขั้นต่อมาเรากดฮอตคีย์ด้วยแป้น F7 เรียกใช้คำสั่ง Run/Trace into เพิ่มเริ่มทดสอบในส่วนอื่นของโปรแกรม

ข้อกำหนดและใช้งาน

Breakpoint

ความสามารถอย่างหนึ่งที่มีให้ใน Integrated Turbo Debugger คือการกำหนดตำแหน่งที่จะให้ระบบหยุดการทำงาน (breakpoint) ได้อย่างสะดวกสบาย ทำให้ผู้ใช้สามารถกำหนดช่วงหรือชุดคำสั่งของโปรแกรมได้เป็นส่วน ๆ ตามต้องการ ทีนี้ลองกลับมาดูตัวอย่างในส่วนการทำงานช่วงแรกของฟังก์ชัน main () จะรอให้ผู้ใช้เติมชื่ออินพุตไฟล์หรือเปิดไฟล์เพื่ออ่านข้อมูล จากการทดสอบเราจะเห็นว่า WORDCNT ได้อ่านและแสดงข้อมูลบรรทัดแรกของอินพุตไฟล์ จนถึงจุดนี้โปรแกรมก็ยังทำงานได้อย่างราบเรียบ คือปัญหาที่เกิดขึ้นไม่ได้อยู่ที่การทำงานของตัวโปรแกรม เพราะฉะนั้นเราสามารถรันผ่านส่วนนี้ของ main () ไปได้ และกำหนดจุดหยุดการทำงานที่ส่วนอื่นของโปรแกรม ซึ่งตำแหน่งที่เราต้องการให้หยุดนั้นจะต้องกำหนดไว้เพื่อให้ดีบักเกอร์รู้ ในที่นี้ให้ใช้แป้น PgDn และปุ่มลูกศร → เพื่อเลื่อนเคอร์เซอร์ให้ไปอยู่ ณ บรรทัดที่มีชุดคำสั่ง While (fgets(... ที่อยู่ต่อจาก Comment ที่เริ่มต้นด้วย Each loop processes...

เมื่อเลื่อนเคอร์เซอร์ไปยังบรรทัด while...แล้ว (จะเห็นว่าบรรทัดไฮไลต์ไม่ได้เลื่อนตามมาด้วย เพราะเรายังไม่ได้สั่งรันโปรแกรม แต่เป็นการเลื่อนหาตำแหน่งด้วยเอดิเตอร์) เราสามารถกำหนดตำแหน่งของ breakpoint ในบรรทัดที่ต้องการ (บรรทัดที่มีเคอร์เซอร์แสดงอยู่) ได้ด้วยการเลือกใช้คำสั่ง Break/Watch/Toggle breakpoint หรืออาจใช้ฮอตคีย์ Ctrl-F8 เมื่อเรียกใช้คำสั่งนี้ ดีบักเกอร์จะแสดงไฮไลต์ที่บรรทัดซึ่งเลือกไว้ โดยจะแสดงตัวอักษรของบรรทัดนั้นให้สว่างกว่าปกติ หมายถึงการกำหนดตำแหน่งของ breakpoint ที่บรรทัดนั้นนั่นเอง ถ้าต้องการจะรันโปรแกรมต่อก็สามารถทำได้ทันทีด้วยการเลือกใช้คำสั่ง Run/Run (หรือใช้ Ctrl-F9 ซึ่งเป็นฮอตคีย์ของคำสั่ง) WORDCNT จะทำงานทันที ในกรณีนี้จะเริ่มทำงานตั้งแต่ต้นโปรแกรมแล้วมาหยุดรอให้ผู้ใช้ใส่ชื่ออินพุตไฟล์ พิมพ์ WORDCNT.DAT แล้วกดแป้น Enter การทำงานของโปรแกรมจะเป็นไปตามปกติทุกอย่างเหมือนกับไม่ได้รันด้วยดีบักเกอร์ เมื่อใส่ชื่ออินพุตไฟล์เสร็จโปรแกรมจะทำงานต่อไปจนกว่าจะพบ breakpoint และจะแสดง Edit window ขึ้นมาแทนที่ จะเห็นว่าเคอร์เซอร์และ execution bar (ไฮไลต์บาร์) จะอยู่ที่บรรทัดเดียวกันคือ while statement ส่วน breakpoint ที่กำหนดไว้ยังคงอยู่ที่นี่ด้วย แต่เราจะมองไม่เห็นแถบแสงเพราะ execution bar ทับเอาไว้ แต่จะเห็นได้อีกก็ต่อเมื่อเลื่อน execution bar ออกไป และในการรันโปรแกรม WORDCNT ระบบจะหยุดที่ตำแหน่งนี้ทุกครั้งจนกว่าจะยกเลิกตำแหน่งนี้

วิธีใช้งาน Ctrl-Break

ถึงแม้ว่าเราจะสามารถกำหนด breakpoint ไว้ที่ตำแหน่งไหนของโปรแกรมก็ได้ บางทีอาจมีกรณีจำเป็นที่จะต้องหยุดการทำงานของโปรแกรมก่อนถึงตำแหน่ง breakpoint ที่กำหนดไว้ ในกรณีนี้สามารถทำได้ด้วยการกดแป้น Ctrl-Break ส่งผลให้โปรแกรมหยุดการประมวลผลทันที หมายความว่าผู้ใช้สามารถสั่งให้โปรแกรมหยุดทำงานได้ตลอดเวลานั่นเอง หลังจากกด Ctrl-Break แล้ว ระบบจะออกจากดีบักเกอร์เข้าสู่เทอร์โบซีเอดิเตอร์ โดยมี execution bar อยู่ตรงบรรทัดที่ระบบจะทำการประมวลผลต่อไปหากได้รับคำสั่งให้รันอีกครั้ง วิธีการทำงานของดีบักเกอร์ก็คือ ตัวดีบักเกอร์จะต้องมีการติดต่อกับดอส ไบออส และส่วนอื่น ๆ (services) ของระบบที่เกี่ยวข้องกับการทำงาน เพื่อให้สามารถรับรู้ได้ว่าคำสั่งที่ได้รับนั้นเป็นดอสรูทีน ไบออสรูทีน หรือของโปรแกรมเอง เพราะฉะนั้นเมื่อผู้ใช้กดแป้น Ctrl-Break ตัวดีบักเกอร์จะหยุดรอจนกระทั่งโปรแกรมกลับไปทำงานที่ค้างอยู่จนถึงตำแหน่งที่เป็นจุดเริ่มต้นของคำสั่งในบรรทัดใหม่ของซีซอร์สโค้ด จึงค่อยหยุดทำงาน มีผลให้ execution bar อยู่ในตำแหน่งนั้นด้วย

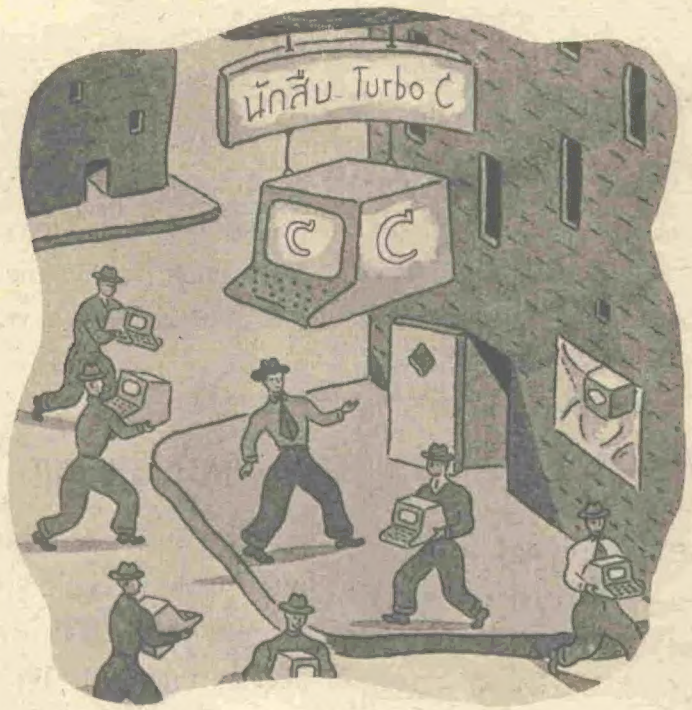
ในกรณีที่ดีบักเกอร์ตรวจพบ Ctrl-Break ตัวที่สองก่อนที่จะทำงานต่อจนถึงคำสั่งสุดท้ายที่ค้างอยู่ และแสดงซอร์สโค้ดของโปรแกรม หมายความว่าจะต้องมีการกด Ctrl-Break ติดต่อกันสองครั้ง ตัวดีบักเกอร์จะเลิกดีบักโปรแกรมนั้นทันที โดยไม่พยายามหาจุดเริ่มต้นของคำสั่งในซอร์สโค้ดชุดต่อไป เป็นผลให้การทำงานของโปรแกรมที่ดีบักนั้นสิ้นสุดลงโดยไม่มีการแสดงเอาต์พุตหรือเรียกฟังก์ชัน exit ใด ๆ เพราะฉะนั้นผู้ใช้ควรกดแป้น Ctrl-Break ติดต่อกันสองครั้งก็ต่อเมื่อโปรแกรมที่กำลังดีบักอยู่นั้นมีการทำงานผิดปกติ เช่น การวนลูปไม่หยุด การกดแป้น Ctrl-Break เพียงครั้งเดียวไม่สามารถหยุดการทำงานได้

แกะรอยโปรแกรม

สำหรับการทดสอบในส่วนของโปรแกรมที่เราคิดว่ามีปัญหาในการทำงานนั้น เราควรจะใช้วิธีการตรวจสอบที่ละเอียด รอบคอบ และให้ความระมัดระวังพอควร วิธีการที่ใช้ในดีบักเกอร์ก็คือการทดสอบผลการทำงานของชุดคำสั่งครั้งละ 1 บรรทัด โดยจะรันและแสดงผลการทำงานของโปรแกรมแต่ละบรรทัด แล้วหยุดเพื่อให้ผู้ใช้สามารถตรวจสอบหรือพิจารณาผลที่ได้รับว่าเป็นไปตามความมุ่งหมายหรือไม่อย่างไร Run/Step over เป็นคำสั่งที่จะใช้เพื่อวัตถุประสงค์นี้ใน Integrated Debugger คอยบอกให้ดีบักเกอร์ทำการรันคำสั่ง while...และอ่านข้อมูลในบรรทัดแรกของอินพุตไฟล์ แล้วเลื่อน execution bar ไปอยู่บนบรรทัดต่อไป (ที่เป็น executable statement) เมื่อเห็นว่าการทำงานของบรรทัดที่ผ่านมาถูกต้องก็ใช้คำสั่ง Run/Step over อีกครั้ง เพื่อให้ดีบักเกอร์รันคำสั่งชุดต่อไป ได้แก่การเรียกใช้ฟังก์ชัน printf

เมื่อต้องการจะดูผลการทำงานของโปรแกรมที่ทำการทดสอบอยู่ว่าให้ผลลัพธ์ที่ถูกต้องตรงความต้องการหรือไม่ ระบบจะแสดงผลไว้ในยูสเซอร์สกรีน ซึ่งเรียกใช้ได้ด้วยคำสั่งที่อยู่ในเมนู Run หรือใช้แป้นฮอตคีย์ Alt-F5 ระบบจะแสดง execution screen ผู้ใช้จะเห็นผลลัพธ์ที่แสดงบนจอภาพได้ หากผลลัพธ์ที่ได้นั้นถูกต้องแสดงว่าคำสั่ง while และ printf ทำงานสำเร็จ ขั้นต่อไปเราจะกลับไปที่ Edit window ด้วยการกดแป้นใด ๆ ก็ได้ คำสั่ง Run/Step over นี้เป็นคำสั่งที่ต้องใช้บ่อยมากในการดีบักโปรแกรม เพราะฉะนั้นบอร์แลนด์จึงได้กำหนดให้มีฮอตคีย์ไว้ใช้เพื่อความสะดวกในการทำงาน และฮอตคีย์สำหรับคำสั่งนี้ก็คือ F8 ทุกครั้งที่กดฟังก์ชันคีย์ตัวนี้ ดีบักเกอร์จะประมวลผลชุดคำสั่งที่อยู่ในตำแหน่งของ execution bar และเลื่อนบาร์นี้ลงไปอีกหนึ่งบรรทัด

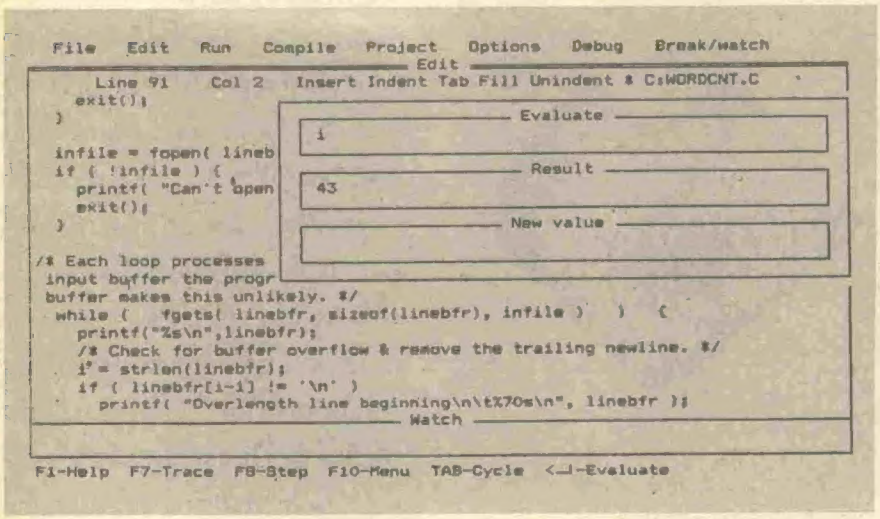
สำหรับการทำงานขั้นต่อไปของตัวอย่าง เราจะใช้ฟังก์ชันคีย์ F8 ประมวลผลคำสั่งชุดต่อไป ซึ่งจะเป็นการคำนวณความยาวของข้อมูลและเก็บค่าที่ได้ไว้ที่ตัวแปร 1 มีข้อควรระวังในการใช้คำสั่ง Trace into หรือ Stepping over ประการหนึ่งคือ ถ้าเป็นการเรียกเข้าไปในไลบรารี่ฟังก์ชัน longjmp ระบบจะไม่หยุดที่บรรทัดต่อไปของคำสั่งในซอร์สโค้ด (เพราะการทำงานของฟังก์ชันนี้ไม่ส่งค่ารีเทอร์นให้) แต่จะทำงานต่อไปจนถึง breakpoint ที่กำหนดไว้หรือจนจบโปรแกรม



ทดสอบ expression

ขีดความสามารถอีกอย่างหนึ่งที่มีให้ใน Integrated Debugger ก็คือการทดสอบการทำงานของ expression ซึ่งผู้ใช้สามารถที่จะกำหนด expression ที่ต้องการทดสอบผลการทำงานให้ดีบักเกอร์ได้ วิธีการแบบนี้จะช่วยให้รู้ได้ว่า expression ที่กำหนดไว้นั้นทำงานได้ถูกต้องตามความต้องการหรือไม่ การเรียกใช้คำสั่งกระทำได้โดยเลือกคำสั่ง Debug/Evaluate ซึ่งจะมีผลให้ดีบักเกอร์เปิดกรอบหน้าต่างบนจอภาพให้ (pop-up window) โดยช่องหน้าต่างนี้จะประกอบด้วย 3 ฟิลด์ที่ทำหน้าที่ต่างกัน ดังรูปที่ 7

- Evaluate field เป็นช่องหน้าต่างที่เปิดให้ผู้ใช้ใส่ expression ที่ต้องการให้ระบบทำการทดสอบ ซึ่งเราสามารถใส่ expression อะไรหรือจะแก้ไขเป็นอย่างอื่นก็ได้ และในกรณีที่ expression ที่ใส่เข้าไปมีความยาวเกินกว่า Expression field box ก็สามารถที่จะเลื่อนจอภาพโดยใช้ปุ่มลูกศร →, ←, Home และ End ได้เหมือนกับที่ใช้ในเอดิเตอร์อื่น ๆ
- Result field เป็นฟิลด์ที่ดีบักเกอร์จะใช้สำหรับแสดงผลลัพธ์ที่ได้จากการประมวลผล expression ที่อยู่ใน Evaluate field
- New Value field เป็นช่องสำหรับให้ผู้ใช้ใส่ค่าใหม่ที่ต้องการ ซึ่งฟิลด์นี้ถือเป็น optional ของคำสั่ง Debug/Evaluate ที่ผู้ใช้จะใช้หรือไม่ก็ได้



รูปที่ 7 การกำหนด expression ในการดีบัก

สำหรับ expression ที่อยู่ใน Evaluated field นั้นจะมีค่าดีฟอลต์เป็นคำ (word) อยู่ที่ตำแหน่งของเคอร์เซอร์ใน Edit window ส่วนในตัวอย่าง เราต้องการรู้ค่าของ i เพราะฉะนั้นให้พิมพ์คำว่า 'i' ลงไปใน Evaluate field และกดแป้น Enter ตัวดีบักเกอร์จะแสดงค่าของ i ใน Result field และค่าที่ได้คือ 43 ซึ่งเป็นค่าที่ถูกต้องตรงตามความเป็นจริง เพราะฉะนั้นเราสามารถกลับไปสู่เมนูหลักได้เลย โดยกดแป้น F10 หรือฮอตคีย์ของระบบ

ชุดคำสั่งกลุ่มต่อไปของโปรแกรม WORDCNT ที่เราจะทำการทดสอบการทำงานได้แก่ กลุ่มที่จะทำหน้าที่ตรวจสอบข้อมูลในบรรทัดที่อ่านเข้ามาจากอินพุตไฟล์ด้วยคำสั่ง fgets เราต้องจบบรรทัดด้วยคำสั่งขึ้นบรรทัดใหม่ ในกรณีที่ไม่มีเครื่องหมายนี้แสดงว่าอินพุตไลน์นั้นมีความยาวเกินกว่าบัฟเฟอร์ที่เตรียมไว้ ระบบจะแสดงข้อความเตือนให้ทราบ แต่ถ้าอินพุตที่ได้จบลงด้วยเครื่องหมายขึ้นบรรทัดใหม่ (newline) ก็แสดงว่าถูกต้อง และโปรแกรมจะทำ

หน้าที่ลบเครื่องหมายนี้ออกไปเอง โดยไม่นับเครื่องหมายนี้เข้าไปในสตริงด้วย

ก่อนที่จะทำการประมวลผล if statement เราควรจะดูด้วยว่าอินพุตที่อ่านเข้ามานั้นเป็นอย่างไร เพื่อจะได้รู้ว่าผลลัพธ์ที่ควรจะได้นั้นคืออะไร วิธีการก็คือเลื่อนเคอร์เซอร์ไปที่คำว่า linebfr ใน Edit Window เลือกคำสั่ง Debug/Evaluate (ใช้แป้นฮอตคีย์ Ctrl-F4 แทนได้) ดีบักเกอร์จะแสดง linebfr ใน Debug/Evaluate window ที่ Evaluate field ให้กดแป้น Enter เพื่อประมวลผลต่อ จะได้ผลลัพธ์ออกมาเป็น

```
To be or not to be: that is the question./n
```

จะเห็นว่ามี '/n' ซึ่งเป็นสัญลักษณ์ newline character อยู่ทางตอนท้ายของประโยคที่ใช้กับเทอร์โบซี

ขั้นต่อไปจะเริ่มรันคำสั่ง if โดยกดแป้น F8 (Run/Step over) เพื่อดูผลการทำงาน จะเห็นว่า execution bar เลื่อนลงไปอยู่ที่ else clause แสดงว่าการทำงานของ if statement เป็นไปอย่างถูกต้อง เราก็ทำการรันชุดคำสั่งใน else clause ต่อ แล้วจึงมาดูผลลัพธ์ใน linebfr อีกที (ใช้คำสั่ง Debug/Evaluate หรือ Ctrl-F4) ซึ่งจะเห็นว่าคำสั่งขึ้นบรรทัดใหม่ (/n) ถูกลบออกไปแล้ว แสดงว่าโปรแกรมยังคงทำงานได้ถูกต้องตามความต้องการ

คำสั่งชุดต่อไปของโปรแกรมก็คือการเรียกใช้ฟังก์ชัน nextword () ที่ทำหน้าที่บอกตำแหน่งของคำ (word) ต่อไปของประโยค (string) ในกรณีตัวอย่างของเราจะเป็นคำแรกของประโยค เมื่อสั่งรันโปรแกรมต่อด้วยการกดแป้น F8 และดูผลการทำงานจากค่าของ wordptr ซึ่งได้จากการทำงานของฟังก์ชัน next word (linebfr) ถ้าโปรแกรมทำงานได้ถูกต้องจะเห็นว่ามีค่าเป็น T อักษรตัวแรกของคำว่า 'To' แสดงว่าการทำงานของโปรแกรมในส่วนนี้ถูกต้อง ขั้นต่อไปโปรแกรมจะทำงานซ้ำอยู่ในลูปจนถึงอักษรตัวสุดท้ายของประโยค ค่าของ wordptr จะต้องมีค่าเป็น null character และการทำงานของลูปก็จะต้องหยุดเพียงเท่านี้

ขณะที่ทำการรันคำสั่ง while นั้น execution bar จะเลื่อนมาอยู่ที่คำสั่งแรกที่อยู่ในลูป นั่นคือการเรียกใช้ฟังก์ชัน wordlen () ทำหน้าที่คำนวณความยาวของแต่ละคำที่ wordptr ชี้อยู่ เมื่อรันคำสั่งนี้แล้วดูค่าของ i (ซึ่งเรียกใช้ wordlen ()) จะเห็นว่ามีค่าเป็น 0 ทั้ง ๆ ที่ค่าที่ถูกต้องควรจะเป็น 2 (เพราะ TO ซึ่งเป็นคำแรกของประโยคนั้นมี 2 ตัวอักษร) แสดงว่าการทำงานของโปรแกรมในส่วนนี้มีข้อความผิดพลาดเกิดขึ้น จัดเป็นบักตัวแรกที่เราพบในโปรแกรม WORDCNT

หยุดคิดกันก่อน

ปกติถ้าโปรแกรมเมอร์พบบักในโปรแกรมของตนเมื่อไรคงจะทำหน้าเบ้ แต่บทความเรื่องนี้เป็นแค่ตัวอย่างจึงไม่ต้องวิตกกังวลนัก ก่อนที่เราจะทำการกำจัดบักออกไป ลองมาดูกันสักนิดว่าการที่โปรแกรมให้ค่าผิด ๆ ออกมาจะมีผลต่อการทำงานของโปรแกรมในช่วงต่อไปอย่างไรบ้าง เมื่อโปรแกรมได้ค่าของ wordlen คืนมาเป็น 0 จะเกิดผลกระทบสองอย่างคือ

- จะทำให้ค่าของ Wordlencnt ซึ่งรับค่าต่อจาก wordlen ()

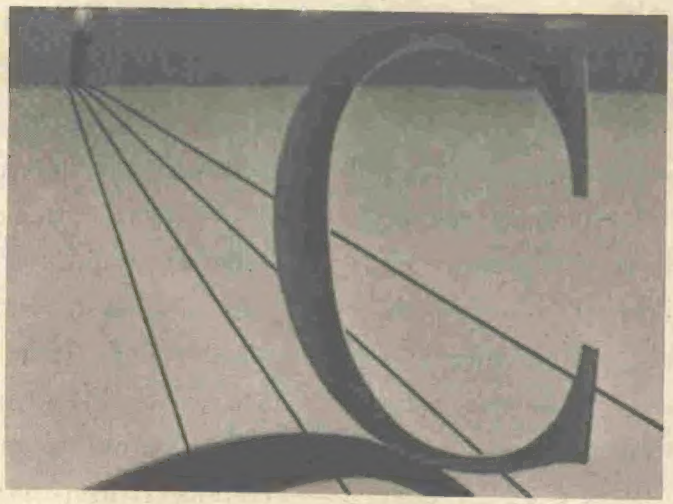
ผิดไปจากความเป็นจริง - จะมีผลให้คำสั่ง wordptr+=i ผิดไปจากที่ควรจะเป็น เพราะจะทำให้ค่าของ wordptr ไม่เปลี่ยนแปลง หมายความว่าการทำงานในรอบที่สองของ while loop เริ่มต้นด้วยค่าของ wordptr เดียวกันกับรอบแรก นั่นคือเมื่อ wordlen คืนค่า 0 มาให้ในการถูกเรียกใช้ครั้งแรก โปรแกรมจะคืนค่า 0 มาให้อีกเช่นเดิมในการเรียกใช้ครั้งต่อมา

เหตุการณ์ที่เกิดขึ้นเช่นนี้แสดงว่ามีบักตัวแรกของ WORDCNT ทำให้ระบบหยุดการทำงาน สาเหตุอาจมาจากความผิดพลาดของโปรแกรมเองหรือไม่ก็ได้ เพราะฉะนั้นสิ่งที่ควรปฏิบัติคือการรันโปรแกรมนี้ใหม่ เพื่อดูว่าจะได้ผลลัพธ์เหมือนเดิมหรือเปลี่ยนแปลง จะได้พยายามหาวิธีการแก้ไขจนกว่าการทำงานของโปรแกรมจะเป็นไปอย่างถูกต้อง

เมื่อจะแก้ไขบัก

ทีนี้เราก็รู้แล้วว่าในโปรแกรมของเรามีบักเกิดขึ้นที่ฟังก์ชัน wordlen นั่นเอง หมายความว่างานของเราจะต้องเริ่มกันที่จุดนี้ แต่ก่อนที่จะลงมือแก้ไข เราลองมาดูขั้นตอนและเครื่องไม้เครื่องมือ (คำสั่งต่าง ๆ) ที่เราจะต้องใช้ในการตรวจสอบและแก้ไขบักว่าจะมีอะไรบ้าง

- ก่อนอื่นเราต้องตรวจสอบให้แน่ใจว่าคำสั่ง Debug/Source debugging และ O/C/C/Obj debug information ถูกกำหนดค่าไว้ให้เป็น On



- เลือกคำสั่ง Compile/Build all เพื่อจัดการให้โปรแกรม WORDCNT ของเราพร้อมที่จะนำมาทำการดีบักได้
- กำหนด breakpoint ของโปรแกรมโดยใช้คำสั่งของเอดิเตอร์ เพื่อเคลื่อนย้ายเคอร์เซอร์ไปตรวจสอบในส่วนของโปรแกรมที่คิดว่ามีข้อผิดพลาดเกิดขึ้น แล้วเลือกคำสั่ง Break/Watch/Toggle breakpoint เพื่อทำการกำหนดตำแหน่งที่ต้องการให้โปรแกรมหยุดทำงาน ใช้คำสั่ง Run/Run (หรือใช้ Ctrl-F9 เพื่อสั่งรันโปรแกรม WORDCNT ตั้งแต่ต้นจนถึง breakpoint
- เลือกใช้คำสั่ง Run/User screen หรือใช้ฮอตคีย์ Alt-F5 เพื่อตรวจดูผลลัพธ์ของโปรแกรมบนยูสเซอร์สกรีน
- สั่งให้ดีบักเกอร์ทำการตรวจสอบการทำงานของโปรแกรมด้วยคำสั่ง Run/Step over (หรือ F8) เพื่อรันโปรแกรมในฟังก์ชัน main () ครั้งละหนึ่งบรรทัด
- เลือกคำสั่ง Debug/Evaluate (ใช้ปุ่มฮอตคีย์ Ctrl-F4) เพื่อสั่งให้ระบบแสดงค่าของตัวแปรต่าง ๆ ที่ใช้
- เมื่อเจอบักต้องพยายามตรวจสอบหาเหตุผลที่ทำให้เกิดบักนั้น เพื่อให้ได้ข้อสรุปและวิธีการที่จะแก้ไขต่อไป

สรุปกันก่อน

สี่เท้ายังรู้พลาด นักปราชญ์ยังรู้พลั้ง โปรแกรมเมอร์บางคนอาจผ่านประสบการณ์เรื่องดีบักมาแล้วอย่างโชกโชน บทความที่ผู้เขียนนำเสนอในคราวนี้คงมีประโยชน์ต่อผู้อ่านที่ต้องคลุกคลีกับการเขียนโปรแกรมด้วยภาษาซีบ้างไม่มากก็น้อย ถึงแม้ว่าจะเป็นเรื่องเก่ามาเล่าใหม่ (อย่าเพิ่งเบื่อ ๆ) ในฉบับหน้ายังมีทีเด็ดแพรวพราวที่น่าสนใจอีกมากมาย คอยติดตามฉบับต่อไปนะครับ